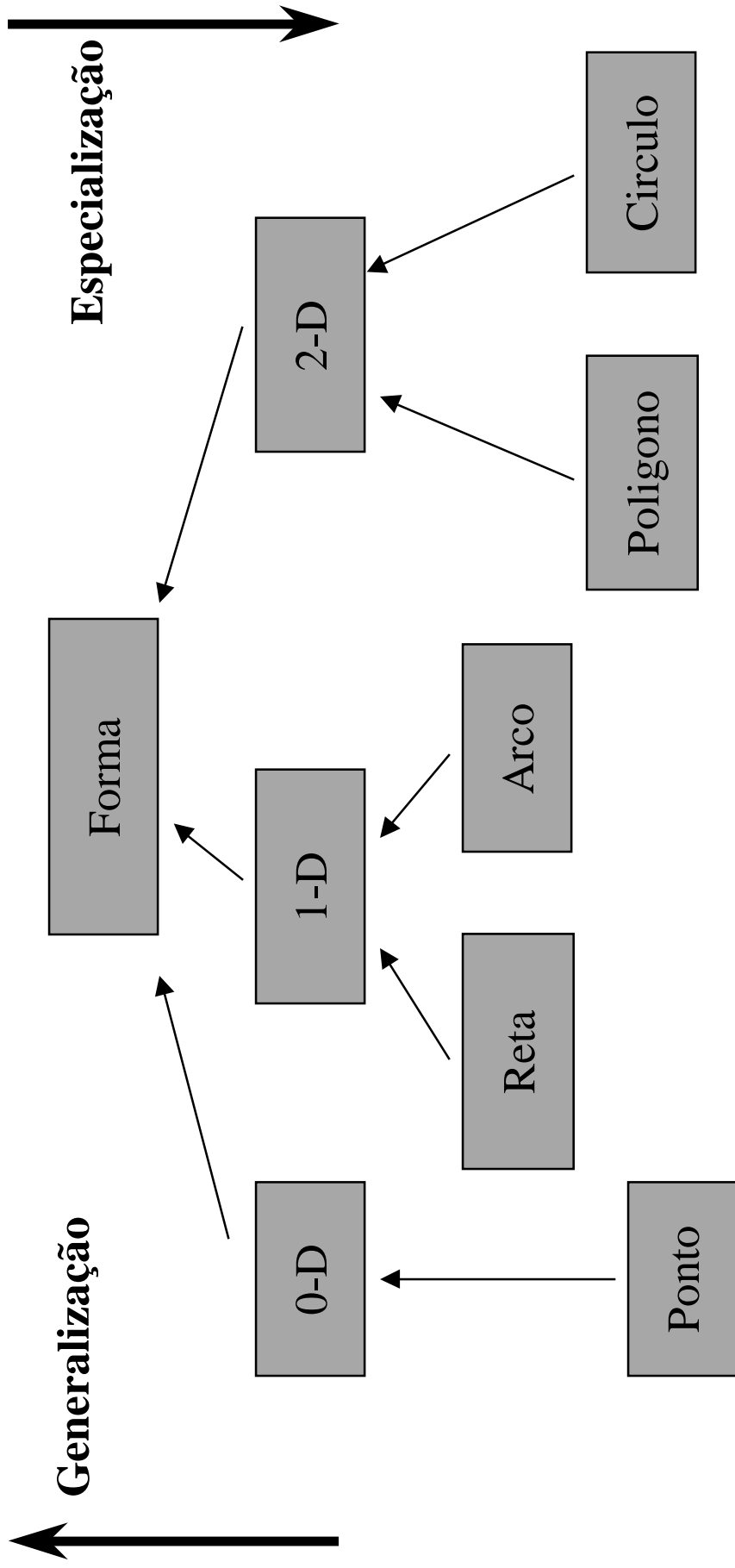


•
•
•

Generalização/Especialização

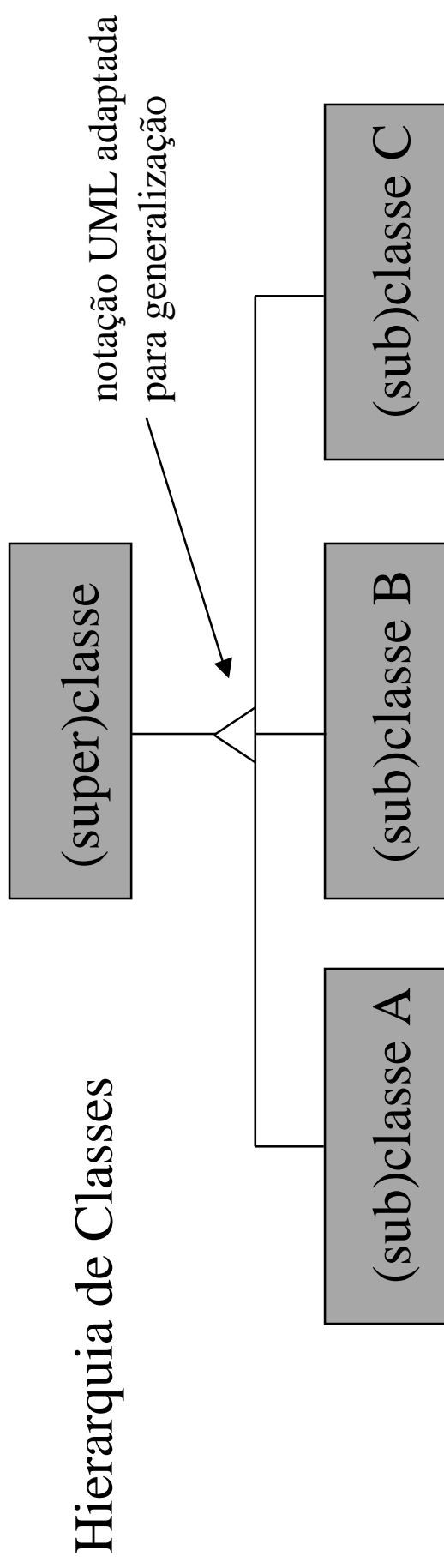
Generalização é a abstração que permite **compartilhar** semelhanças, preservando diferenças



• • • • •

- # Notação para Generalização

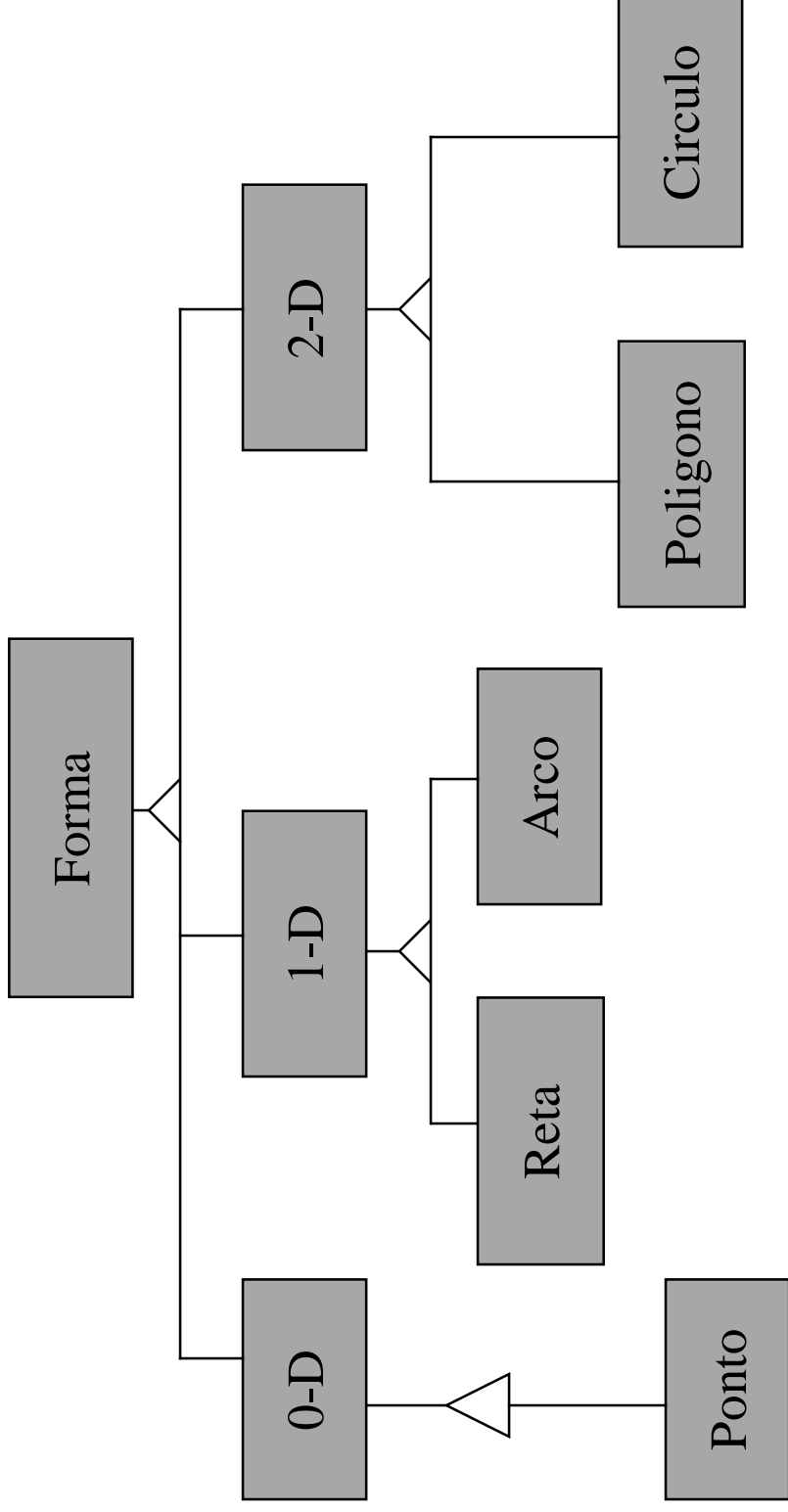
Notação para Generalização



Classes derivadas ou subclasses

•
•
•

Hierarquia de classes - Exemplo



Hierarquia de Classes de Formas Geométricas

• • • • • • • • • •

- # Herança

Herança

Uma classe derivada **herda** a estrutura de dados e métodos de sua classe “base”, mas pode seletivamente:

- adicionar novos métodos
- estender a estrutura de dados
- redefinir a implementação de métodos já existentes

Uma classe “base” proporciona a funcionalidade que é comum a todas as suas classes derivadas, enquanto que uma classe derivada proporciona a funcionalidade adicional que especializa seu comportamento.

•
•
•

Herança em Java

```
public class Circulo {.....}

public class CirculoColorido extends Circulo {
    public string cor;

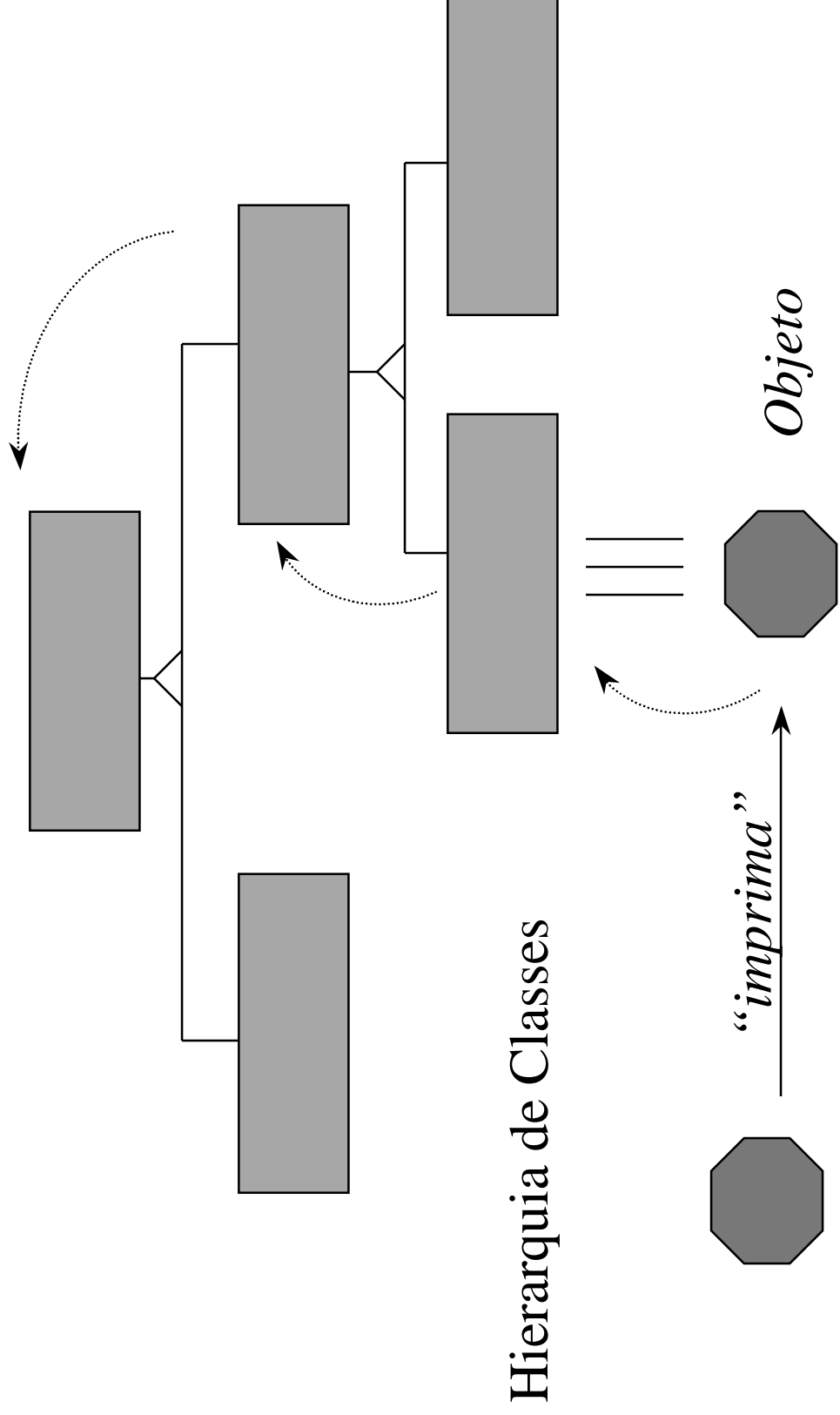
    public void mostra() {.....}

    public void mudaCor(novacor) { cor = novacor; }
}
```

• • • • •

•
•
•

Métodos em uma Hierarquia



• • • • • • • • • •

•
•
•

Resumindo...

- ✓ Tudo é um **objeto**
- ✓ A computação é executada por objetos que se **comunicam**
- ✓ Todo objeto é uma instância de uma **classe**
- ✓ A classe é o repositório do **comportamento** associado com um objeto
- ✓ Classes são organizadas em uma **estrutura de árvore** com uma única raiz

• • • • • • • • • •

- # Atributos e Métodos de Classe



- # Atributos e Métodos de Classe

- métodos de classe somente podem ser invocados em uma classe
- métodos de objetos somente podem ser invocados em um objeto

- há apenas uma cópia de um atributo de classe referente a todos os objetos desta classe

- # Interface

Interface

Uma interface em Java é a descrição de um comportamento.

```
public interface Forma {
    public void mostra();
    public float calculaArea();
}

public classe Circulo implements Forma {
    public float raio, x, y;
    ....
    public void mostra(){.....}
    public float calculaArea(){.....}
}
```

É escrita de uma maneira que se parece com a definição de uma classe. No entanto, não há implementações associadas com métodos.

- # Herança por Especialização

Circulo	
---------	--

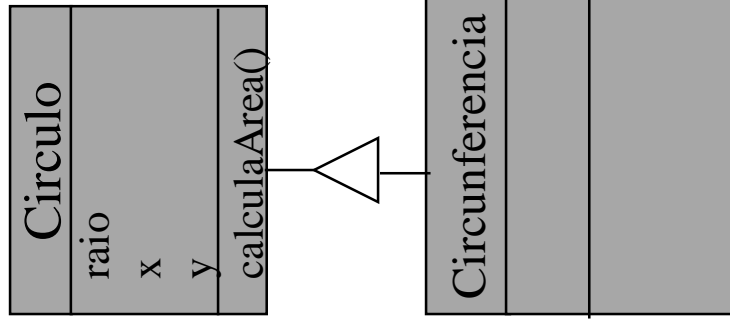


- | |
|-----------------|
| CirculoColorido |
| cor |

-
-
-

Herança por Construção

A classe filha faz uso do comportamento fornecido pela classe pai
mas não é um subtipo da classe pai.

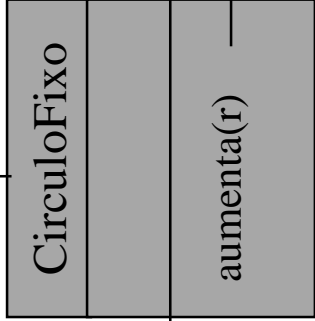


Instâncias da classe `Circunferencia`
respondem ao método `calculaArea()` !!

-
-
-
-
-
-
-
-

- # Herança por Limitação

Circulo
raio
x
y
aumenta(r)



-

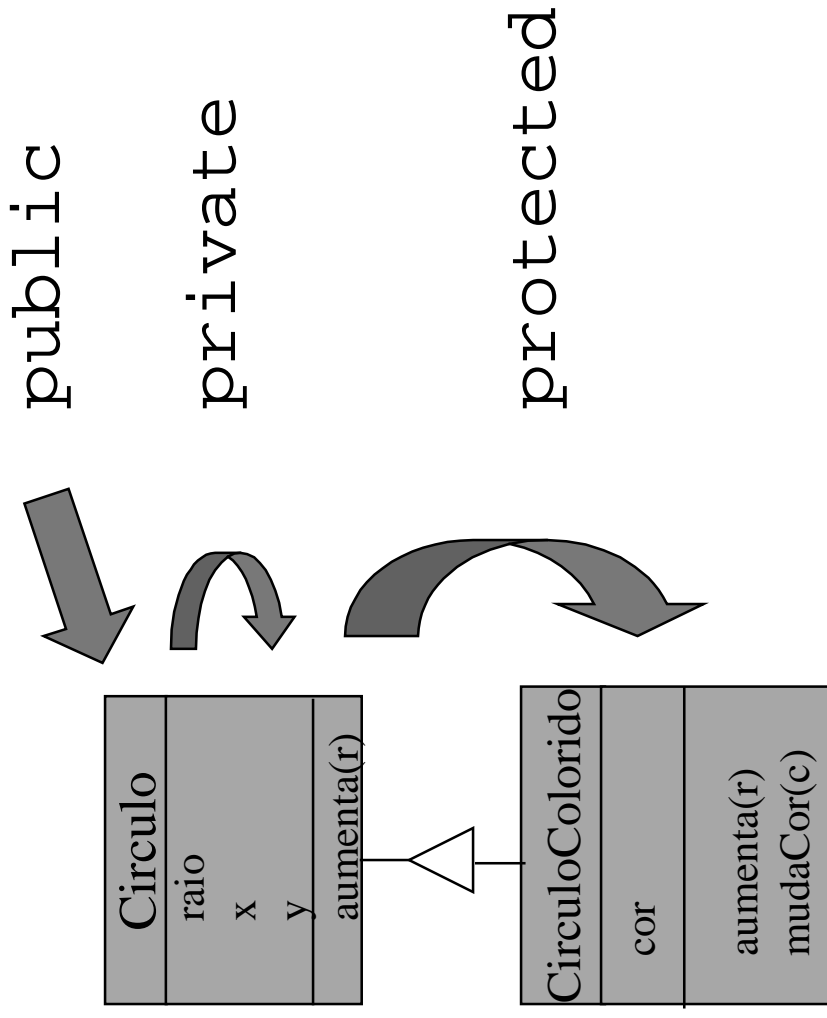
- # Herança por Combinação



```
moveAoRedor(a)
```

•
•
•

Encapsulamento e Herança em Java



Níveis de encapsulamento de atributos e métodos de uma classe

• • • • • • • • • •

•
•
•

Benefícios do uso de Herança

- ✓ Reutilização de Software
- ✓ Maior confiabilidade
- ✓ Compartilhamento de código
- ✓ Consistência de interface

• • • • • • • • • •

•
•
•

Benefícios do uso de Herança

- ✓ Componentes de software
- ✓ Prototipação rápida
- ✓ Ocultação de informação

• • • • • • • • • •

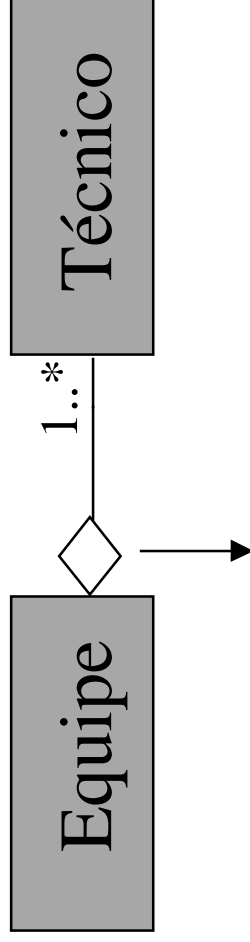
•
•
•

Composição e Agregação

Um objeto pode ser composto a partir de outros objetos e isso é descrito utilizando-se relacionamentos.



Uma empresa é formada de divisões operacionais

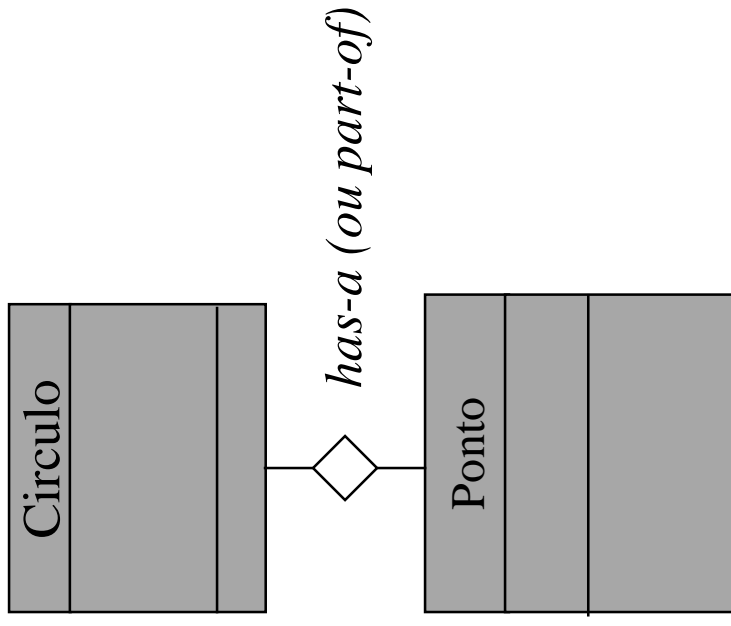
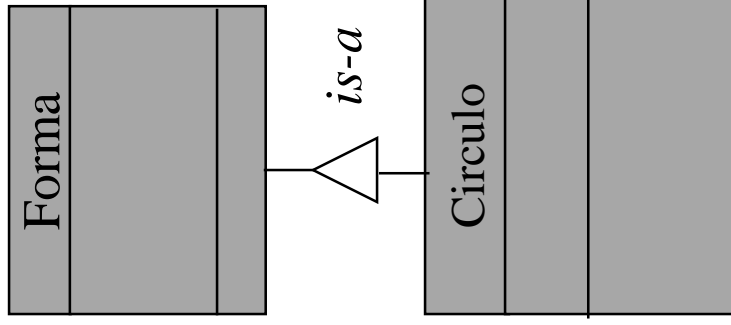


Uma equipe é formada por técnicos

• • • • •

•
•
•

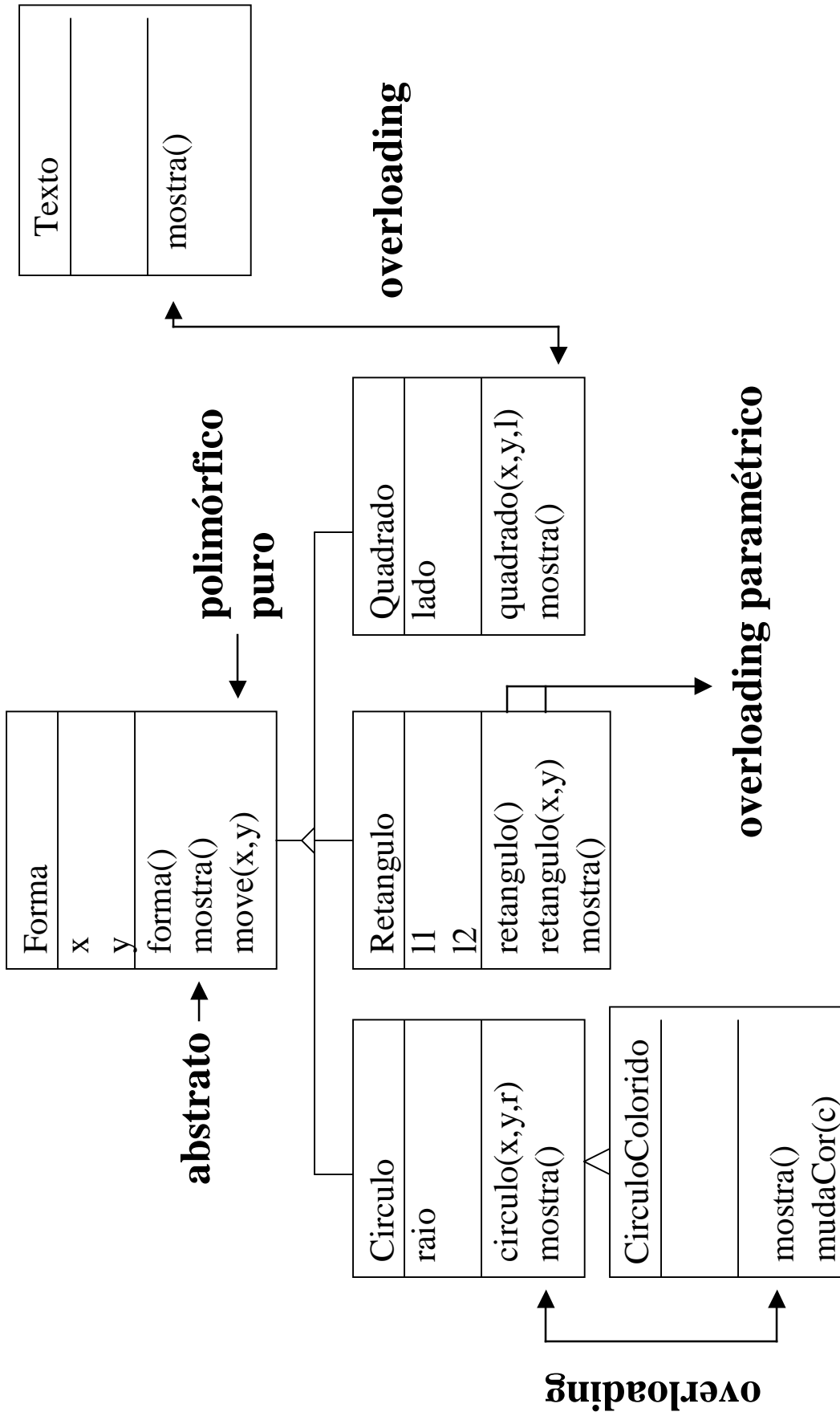
Composição e Herança



• • • • • • • • • •

•
•
•

Polimorfismo



•

•

•

•

•

•

•

•

•

•

•

•

•

•

•